# Machine Learning

## Clustering

Claudio Sartori

DISI
Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

# The problem of clustering

For a comprehensive review see [Jain et al.(1999)Jain, Murty, and Flynn]

- Given – a set of $N$ objects $x_i$, each described by $D$ values $x_{id}$
- Task – find a *natural* partitioning in $K$ clusters and, possibly, a number of *noise* objects
- Result – a *clustering scheme*, i.e. a function mapping each data object to the sequence $[1 \dots K]$ (or to noise)

Desired property of clusters

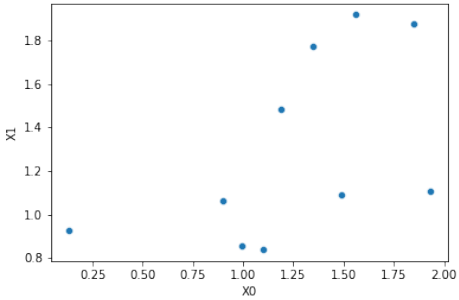
- objects in the same cluster are similar
  - look for a clustering scheme which maximizes intra–cluster similarity

# A little bit of formality

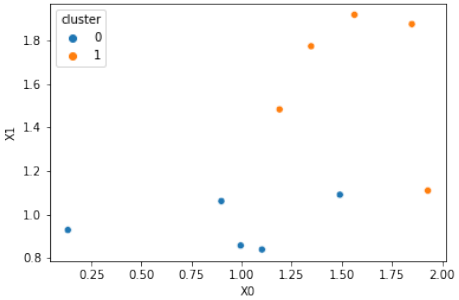
- find a function $clust()$ from $\mathcal{X}$ to $1..K$ such that:
  - $\forall x_1, x_2 \in \mathcal{X}$, $clust(x_1) = clust(x_2)$ when $x_1$ and $x_2$ are *similar*
  - $\forall x_1, x_2 \in \mathcal{X}$, $clust(x_1) \neq clust(x_2)$ when $x_1$ and $x_2$ are *not similar*

# Clustering

2-d Data



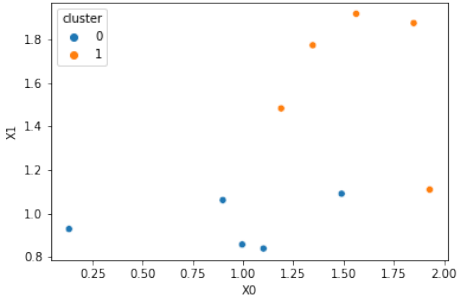

2-d Data clustered



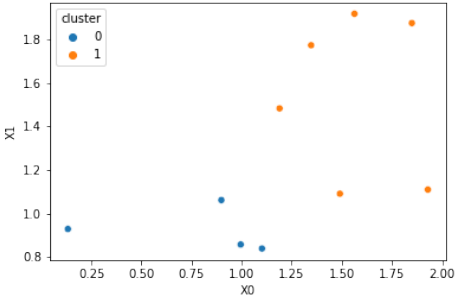

The clustering function maps points to clusters

# Clustering - two different clustering functions

$clust^a$




$clust^b$




Which of the two is better, i.e. maximises intra–cluster similarity?

A measure is needed

# Ideas for a measure?

- the sum of the distances between a point and the others in the same cluster?
  - 
- the average of the distances between a point and the others in the same cluster?
  - 
- the sum of the squared distances?
  - 
- we could choose a point which, in some sense, represents the points of the cluster
  - 
- ...

# Ideas for a measure? Discussion

- the sum of the distances between a point and the others in the same cluster?
  - bigger for bigger clusters
- the average of the distances between a point and the others in the same cluster?
  - not influenced by cluster size
- the sum of the squared distances?
  - more penalisation for sparsity
- we could choose a point which, in some sense, represents the points of the cluster
  - what about the average of the coordinates?
- ...

# Centroid

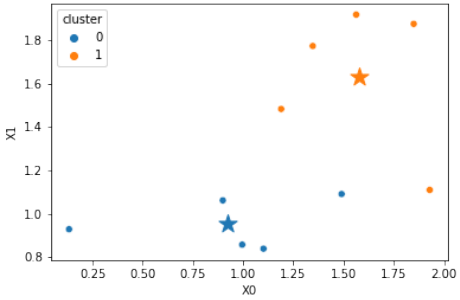
- a point with coordinates computed as the average of the coordinates of all the points in the cluster
- in physics it is the center of gravity of a set of points of equal mass
- for each cluster $k$ and dimension $d$, the $d$ coordinate of the centroid is

$$centroid_d^k = \frac{1}{|x_i : clust(x_i) = k|} \sum_{x_i : clust(x_i) = k} x_{id}$$

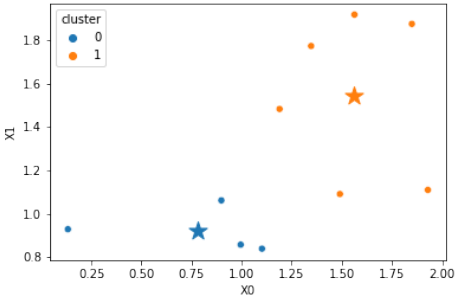
# Clustering - two different clustering functions

The stars represent the *centroids*

$clust^a$




$clust^b$




Which of the two is better, i.e. maximises intra–cluster similarity?

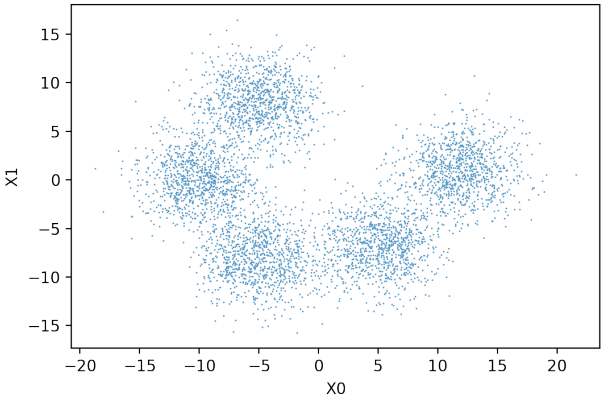
A measure is needed

# Taxonomy of the clustering methods

- Partitioning
  - K-means (MacQueen 67), expectation maximization (Lauritzen 95), CLARANS (Ng and Han 94)
- Hierarchic
  - agglomerative/divisive, BIRCH (Zhang et al 96), CURE (Guha et al 98)
- Based on linkage
- Based on density
  - DBSCAN (Ester et al 96), DENCLUE (Hinnenburg and Keim 98)
- Statistics
  - IBM-IM demographic clustering, COBWEB (Fisher 87), Autoclass (Cheeseman 96)

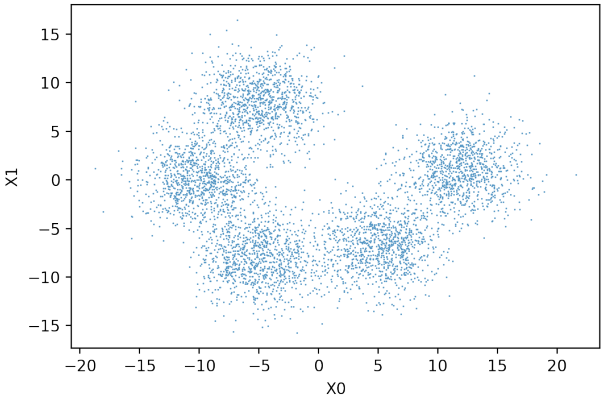
## Some data

- Could be modeled as a five component gaussian mixture
  - *. . . statistician voice. . .*
- How do we guess the number five in a $D$–dimensional space (with $D \geqslant 2$)?



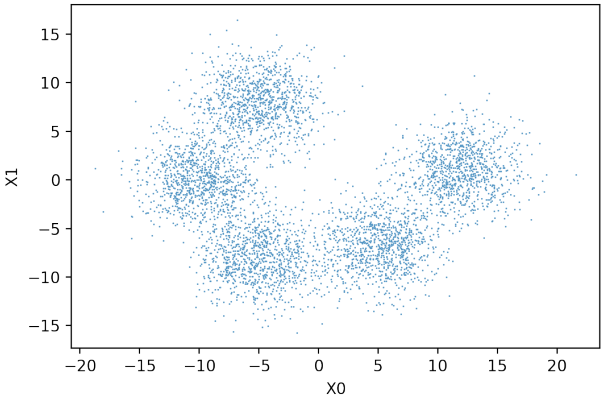
## Transmission

- Transmit the coordinates of points
- Allow only two bits per point
  - the transmission will be lossy
- Need a coding/decoding mechanism



## How much loss?

- Sum of the squared errors between the real points and their encoding/decoding
- Which encoding/decoding minimizes the loss?

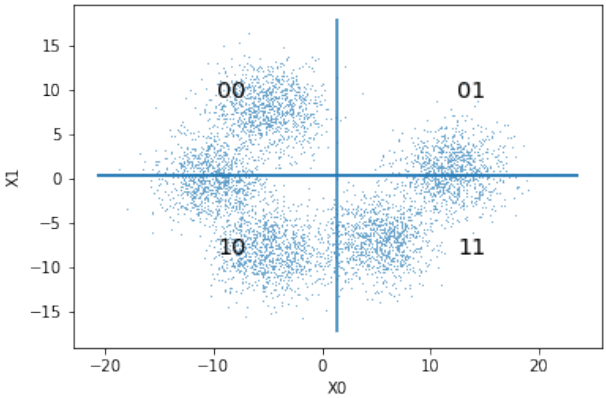


## How much loss?

- Sum of the squared errors between the real points and their encoding/decoding

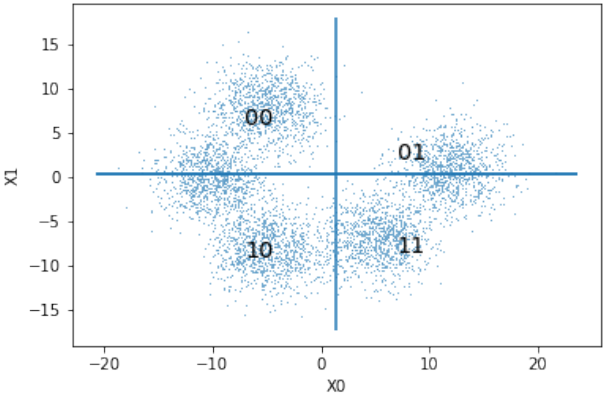
## First idea

- partition the space into a grid of cells
- decode each pair of bits with the center of the grid cell



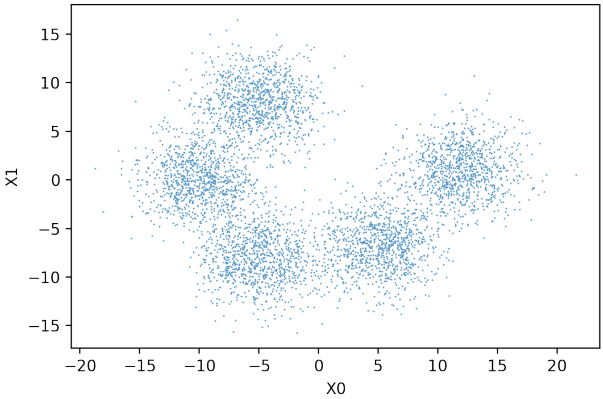
## Improvement

- partition the space into a grid of cells
- decode each pair of bits with the centeroid of the points in the grid cell



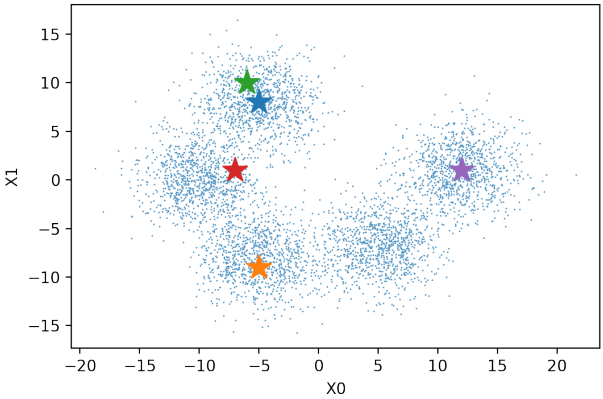
## K–means

1. Ask the user the number of clusters $K$
   1.1 $\Rightarrow 5$



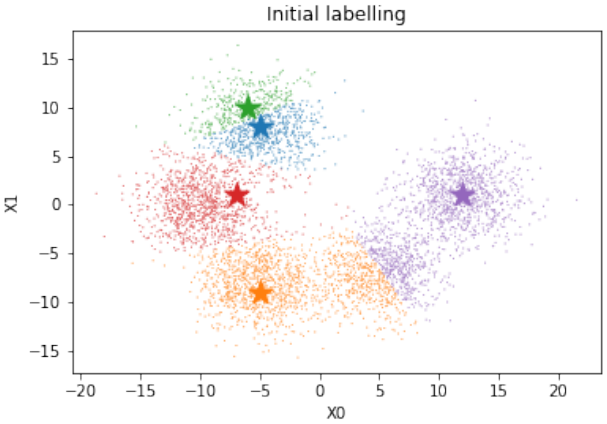
## K–means

1. Ask the user the number of clusters $K$
2. Random choice of $K$ points as temporary centers



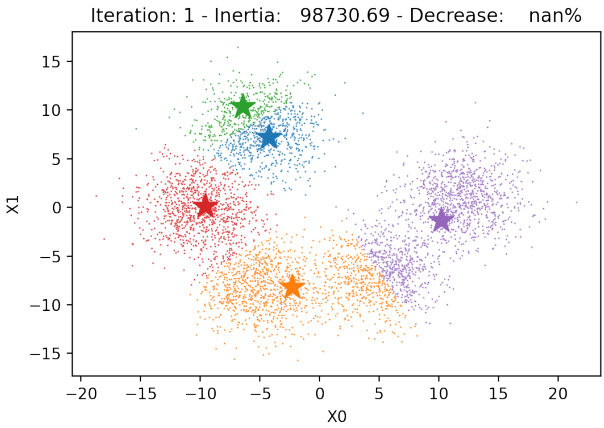
## K–means

1. Ask the user the number of clusters $K$
2. Random choice of $K$ points as temporary centers
3. Each point finds his nearest center and is labelled (i.e. colored) accordingly

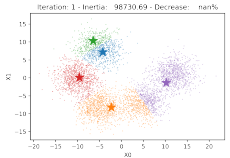


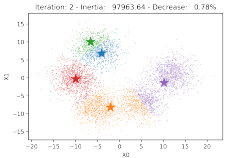
# K–means

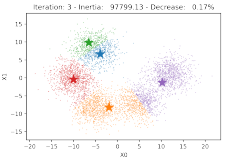
1. Ask the user the number of clusters $K$
2. Random choice of $K$ points as temporary centers
3. Each point finds his nearest center
4. for each center finds the centroid of its points . . .
5. . . . and move there the center

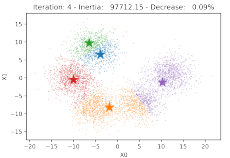


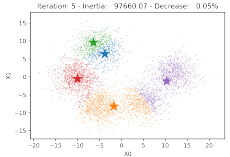
# KMeans working . . .

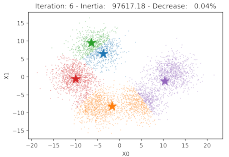


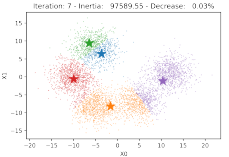



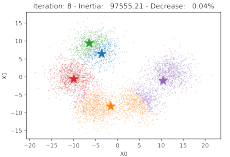



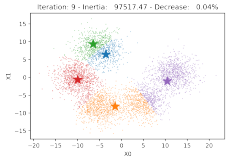



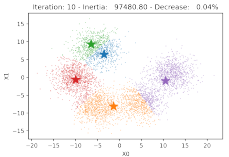



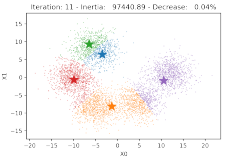



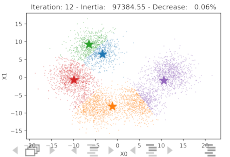



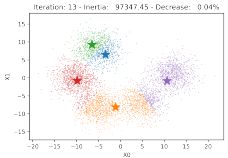



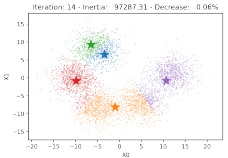



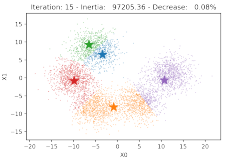



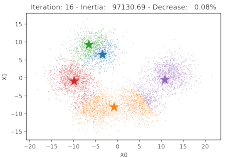



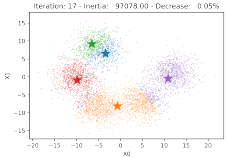
# KMeans working . . .

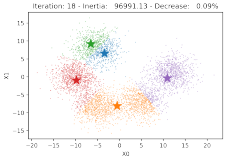


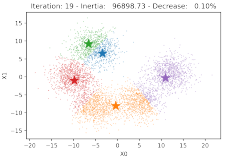



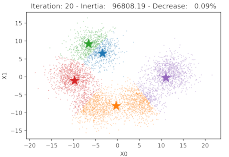



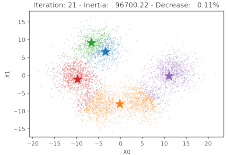



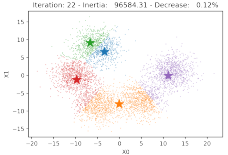



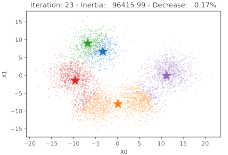



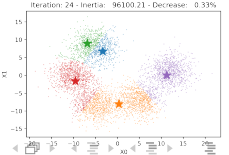



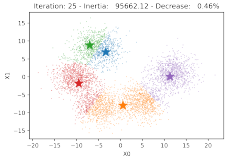



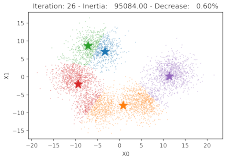



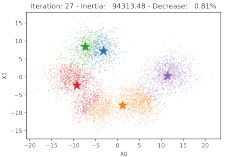



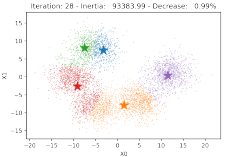



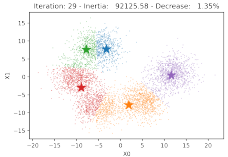
# KMeans working . . .

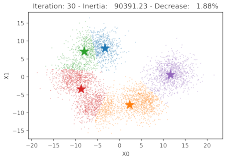


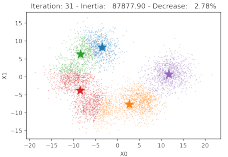



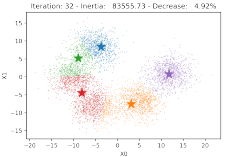



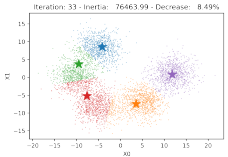



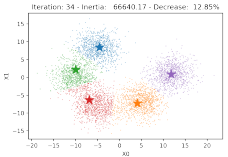



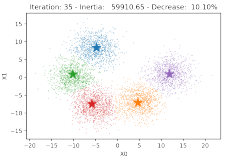



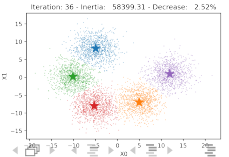



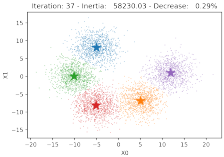



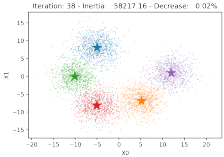



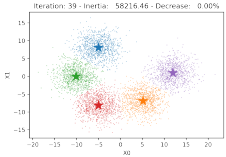



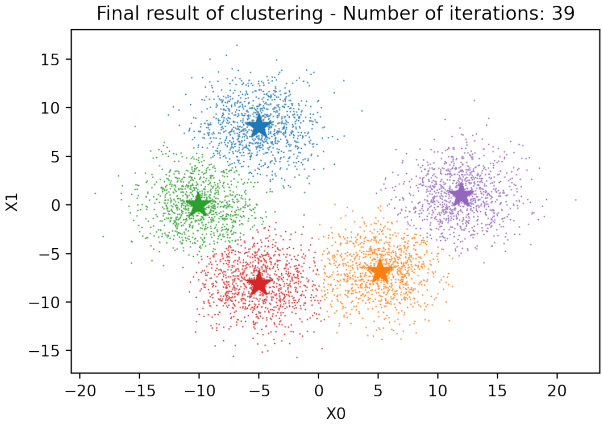
# KMeans working . . .



K–means ends


Final result of clustering - Number of iterations: 39
X1
X0
15
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
15
20

# Questions

1. What are we trying to optimize?
2. Is termination guaranteed?
3. Are we sure that the best clustering scheme is found?
   3.1 Which is the definition of best clustering scheme?
4. How should we start?
5. How can we find the number of clusters?

# Question 1: Distortion

Frequetly called in the literature **Inertia**

Given:

| | |
|---|---|
| a dataset | $\{x_i, i = 1 \dots N\}$ |
| a coding function | $\text{Encode} : \{R\}\hat{}D \rightarrow [1..K]$ |
| a decoding function | $\text{Decode} : [1..K] \rightarrow \mathbb{R}^D$ |
| define | $\text{Distortion} = \sum_{i=1}^{N} (x_i - \text{Decode}(\text{Encode}(x_i)))^2$ |
| shortcut | $\text{Decode}(k) = \mathbf{c}_k$ |

then

$$\text{Distortion} = \sum_{i=1}^{N} (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

# Minimal distortion I

$$\text{Distortion} = \sum_{i=1}^{N} \left(x_i - \mathbf{c}_{\text{Encode}(x_i)}\right)^2$$

Which properties are requested to $\mathbf{c}_1, \ldots, \mathbf{c}_K$ for the minimal distortion?

1. $x_i$ must be encoded with the nearest center

> Why?
> Because otherwise the distortion could be reduced by substituting $\text{Encode}(x_i)$ with the nearest center

$$\mathbf{c}_{\text{Encode}(x_i)} = \underset{\mathbf{c}_j \in \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}}{\text{argmin}} \left(x_i - \mathbf{c}_j\right)^2$$

# Minimal distortion II

$$\text{Distortion} = \sum_{i=1}^{N} \left(x_i - \mathbf{c}_{\text{Encode}(x_i)}\right)^2$$

Which properties are requested to $\mathbf{c}_1, \ldots, \mathbf{c}_K$ for the minimal distortion?
2. The partial derivative of distortion w.r.t. the position of each center must be zero

Why?
Because in that case the function has either a maximum or a minimum

## Step 2. The partial derivative of distortion w.r.t. the position of each center must be zero

$$\begin{aligned}
\text{Distortion} &= \sum_{i=1}^{N} \left(x_i - \mathbf{c}_{\text{Encode}(x_i)}\right)^2 \\
&= \sum_{j=1}^{K} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} \left(x_i - \mathbf{c}_j\right)^2 \\
\frac{\partial\, \text{Distortion}}{\partial \mathbf{c}_j} &= \frac{\partial}{\partial \mathbf{c}_j} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} \left(x_i - \mathbf{c}_j\right)^2 \\
&= -2 \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} \left(x_i - \mathbf{c}_j\right)
\end{aligned}$$

## 2. The partial derivative of distortion w.r.t. the position of each center must be zero

When distortion is minimal

$$\mathbf{c}_j = \frac{1}{|\,\text{OwnedBy}(\mathbf{c}_j)|} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} x_i$$

# Minimal distortion III

$$\text{Distortion} = \sum_{i=1}^{N} \left(x_i - \mathbf{c}_{\text{Encode}(x_i)}\right)^2$$

Which properties are requested to $\mathbf{c}_1, \ldots, \mathbf{c}_K$ for the minimal distortion?

1. $x_i$ must be encoded with the nearest center
2. each center must be the centroid of the points it owns

# Algorithm: Improving a sub–optimal solution

$$\text{Distortion} = \sum_{i=1}^{N} (x_i - \mathbf{c}_{\text{Encode}(x_i)})^2$$

Which properties are requested to $\mathbf{c}_1, \ldots, \mathbf{c}_K$ for the minimal distortion?

1. $x_i$ must be encoded with the nearest center
2. each center must be the centroid of the points it owns

$\Rightarrow$ Alternately perform steps 1 and 2

It can be proven that after a finite number of steps the system reaches a state where neither of the two operations changes the state

Why?

# Question 2: Algorithm termination

- There is only a finite number of ways to partition N objects into K groups
- The state of the algorithm is given by the two encode/decode functions
- The number of configurations where all the centers are the centroids of the points they own is finite
- If after one iteration the state changes, the distortion is reduced
- Therefore each change of state bring to a state which was never visited before
- In summary, sooner or later the algorithm will stop because there are no new states reachable

# Question 3: Local or global minimum?

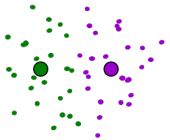
Is the ending state the best possible?

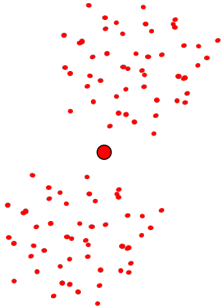
- Not necessarily
- An example

# Question 3: Local or global minimum?

Is the ending state the best possible?

- Not necessarily
- An example





# Question 4: Looking for a good ending state

- The starting point is important
  - choose randomly the first starting point
  - choose in sequence the $2..K$ starting points as far as possibile from the preceding ones
- Re-run the algorithm with different starting points

# Question 5: Choose the number of clusters

not so easy...

- try various values
- use a quantitative evaluation of the quality of the clustering scheme to decide among the different values
- the best value finds the optimal compromise between the minimization of intra-cluster distances and the maximization of the inter cluster distances

# The proximity function

- The most obvious solution, used in the previous formulas is the euclidean distance
  - good choice, in general, for vector spaces
- Several alternative solutions for specific data types and data sets
  - see the "Data" module for additional discussions

# Sum of Squared Errors I

The official name of the distortion

$$\begin{aligned} \text{SSE} &= \sum_{j=1}^{K} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2 \\ &= \sum_{j=1}^{K} \text{SSE}_j \end{aligned}$$

# Sum of Squared Errors II

- A cluster $j$ with high $SSE_j$ has low quality
- $SSE_j = 0$ if and only if all the points are coincident with the centroid
- $SSE$ decreases for increasing $K$, is zero when $K = N$
- $\Rightarrow$ minimizing $SSE$ is not a viable solution to choose the best $K$
  - more discussions on this in the section on "Evaluation of the quality of a clustering scheme"

# Empty clusters

- It may happen, at some step, that a centroid does not own any point
- This changes the initial requirement of K clusters
- $\Rightarrow$ choose a new centroid
  - choose a point far away from the empty centroid or
  - choose as new centroid a random point in the cluster with the maximum *SSE*
    - in this way the cluster with the lowest quality will be split in two

# Outliers

- are points with high distance from their centroid
  - high contribution to *SSE*
- have a bad influence on the clustering results
  - sometimes it is a good idea to remove them
  - the choice is related to the application domain

# Common uses of K–means

- It can be easily used in the beginning, for the esploration of data
- In a one-dimension space it is a good way to discretize the values of a domain in non-uniform buckets
- It is the basis for vector quantization, a classical technique for signal processing and compression
- Used for choosing the color palettes
  - gif compressed images: color quantization

# Complexity

Given:

- $T$ number of iterations
- $K$ number of clusters
- $N$ number of data points
- $D$ number of dimensions

the time complexity is

$$\mathcal{O}(TKND)$$

# Pros and cons of K–means

- Strong points
  - fairly efficient, nearly linear in the number of data points
    - in general $T, K, D << N$
- Weak points
  - in essence it is defined for spaces where the centroid can be computed
    - e.g. when the Euclidean distance is available, also other distance functions work well
    - cannot work with nominal data
  - requires the K parameter
    - nevertheless the best K can be found with iterations
  - it is very sensitive to outliers
  - does not deal with noise
  - does not deal properly with non convex clusters

# Evaluation of a clustering scheme

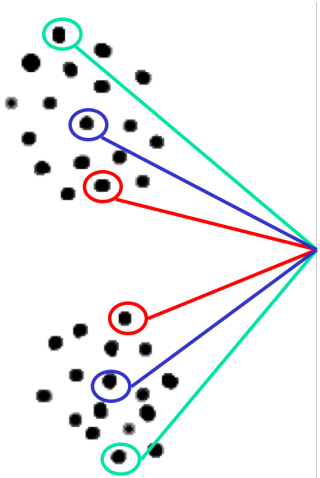
- It is related only to the result, not to the clustering technique
- Clustering is a non supervised method
  - the evaluation is critical, because whenever there is very little apriori information, such as class labels
  - we need one or more score function to measure various properties of the clusters and of the clustering scheme as a whole
    - in the literature the words *score* and *index* are considered synonyms in this context
  - if some supervised data are available, they can be used to evaluate the clustering scheme
- In 2D the clusters can be examined visually
- In higher order spaces the 2D projections can help, but in general it is better to use more formal methods

# Issues on the evaluation of clustering

- Distinguish patterns from random apparent regularities
- Find the best number of clusters
- Non supervised evaluation
- Supervised evaluation
- Relative comparison of clustering schemes

# Measurement criteria

- Cohesion
  - proximity of objects in the same cluster should be high
- Separation between two clusters
  - how to measure it? several choices
    - distance between the nearest objects in the two clusters
    - distance between the most distant objects in the two clusters
    - distance between the centroids of the two clusters

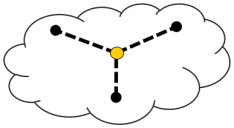


# Proximity and others

- Similarity – Proximity
  - a two variable function measuring how much two objects are similar, according to the values of their properties
- Dissimilarity
  - a two variable function measuring how much two objects are different, according to the values of their properties
  - e.g. the Euclidean distance

# Cohesion – Prototype based

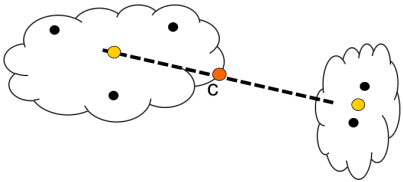
- The sum of the proximities between the elements of the cluster and the geometric center (the prototype)
  - *centroid*
    - a point in the space whose coordinates are the means of the dataset
  - *medoid*
    - an element of the dataset whose average dissimilarity with all the elements of the cluster is minimal
    - not necessarily unique, used in contexts where the mean is not defined, e.g. 3D trajectories or gene expressions



$$\mathrm{Coh}(k_i) = \sum_{x \in k_i} \mathrm{Prox}(x, \mathbf{c}_i)$$

# Separation – Prototype based[1]

- Separation between two clusters
  - proximity between the prototypes




$$\text{Sep}(k_i, k_j) = \text{Prox}(\mathbf{c}_i, \mathbf{c}_j)$$

1 It is the blue case in page 51

# Global separation of a clustering scheme

## SSB – Sum of Squares Between clusters

$$\mathbf{c} = \text{global centroid of the dataset}$$

$$\text{SSB} = \sum_{i=1}^{K} N_i \, \text{Dist}(\mathbf{c}_i, \mathbf{c})^2$$

# Link between cohesion and separation – I

- TSS = Total Sum of Squares
  - sum of squared distances of the points from the global centroid
- TSS = SSE + SSB
- the total sum of squares is a global property of the dataset, independent from the clustering scheme
- for a given dataset, minimise SSE $\Leftrightarrow$ maximise SSB

# Link between cohesion and separation – II

$$\begin{aligned}
\mathrm{TSS} &= \sum_{i=1}^{K}\sum_{x\in k_i}(x-\mathbf{c})^2 = \sum_{i=1}^{K}\sum_{x\in k_i}((x-\mathbf{c}_i)-(\mathbf{c}-\mathbf{c}_i))^2 \\
&= \sum_{i=1}^{K}\sum_{x\in k_i}(x-\mathbf{c}_i)^2 - 2\sum_{i=1}^{K}\sum_{x\in k_i}(x-\mathbf{c}_i)(\mathbf{c}-\mathbf{c}_i) + \sum_{i=1}^{K}\sum_{x\in k_i}(\mathbf{c}-\mathbf{c}_i)^2 \\
&= \sum_{i=1}^{K}\sum_{x\in k_i}(x-\mathbf{c}_i)^2 + \sum_{i=1}^{K}\sum_{x\in k_i}(\mathbf{c}-\mathbf{c}_i)^2 \\
&= \sum_{i=1}^{K}\sum_{x\in k_i}(x-\mathbf{c}_i)^2 + \sum_{i=1}^{K}|k_i|(\mathbf{c}-\mathbf{c}_i)^2 = \mathrm{SSE}+\mathrm{SSB}
\end{aligned}$$

since $\sum_{x\in k_i}(x-\mathbf{c}_i) = 0$ by definition of $\mathbf{c}_i$

# Evaluation of specific clusters and objects

- Each cluster can have its own evaluation
  - the worst clusters can be considered for additional split
- A weakly separated pair of clusters could be considered for merging
- Single objects can give negative contribution to the cohesion of a cluster or to the separation between two clusters
  - border objects

# Silhouette score of a cluster – I

Requirements for a clustering quality score

- values are in a standard range, e.g. $-1, 1$
- increases with the separation between clusters
- decreases for clusters with low *cohesion*

$$\begin{aligned} \text{SSE} &= \sum_{j=1}^{K} \sum_{i \in \text{OwnedBy}(\mathbf{c}_j)} (x_i - \mathbf{c}_j)^2 \\ &= \sum_{j=1}^{K} \text{SSE}_j \end{aligned}$$

# Silhouette score of a cluster – I

Requirements for a clustering quality score

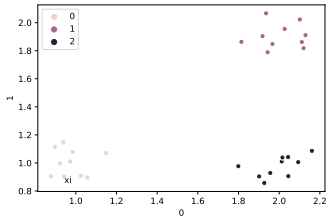
- values are in a standard range, e.g. $-1, 1$
- increases with the separation between clusters
- decreases for clusters with low *cohesion*, or, in other words, with high *sparsity*

# Silhouette score of a cluster – II

Consider the individual contribution of each object, say $x_i$

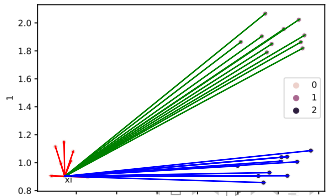
Contribution to cluster sparsity: the average of red distances

$$a_i = \underset{j, y(x_j)=y(x_i)}{\text{average}} \; dist(x_i, x_j)$$




Contribution to separation from other clusters: the minimum of the two averages of green and blue distances

$$b_i = \min_{k \in \mathcal{Y}, k \neq y(x_i)} \left( \underset{j, y(x_j)=k}{\text{average}} \, dist(x_i, x_j) \right)$$



# Silhouette score of a cluster – III

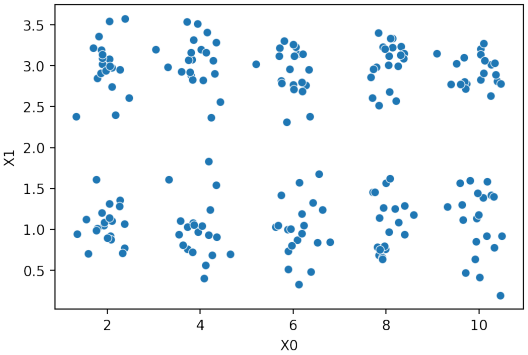
- Silhouette score of $x_i$

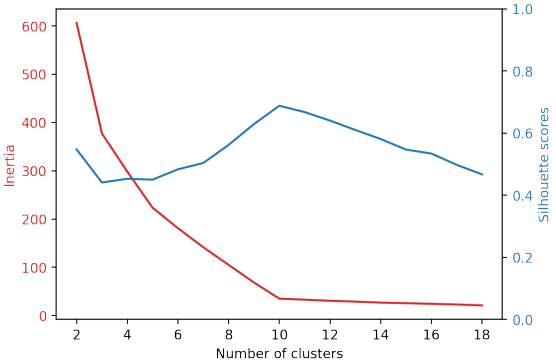
$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1]$$

- For the global score of a cluster/clustering scheme compute the average score over the cluster/dataset
- Intuition
  - when the score is less than zero for an object it means that there is a dominance of objects in other clusters at a distance smaller than objects of the same cluster

# Example: Inertia and silhouette scores

## Testing K-means with different numbers of clusters



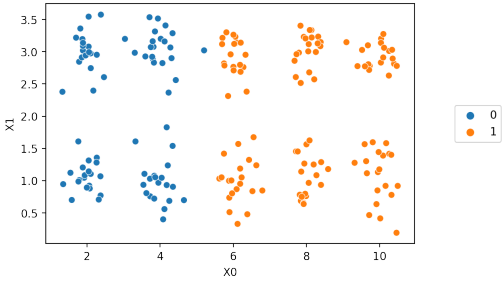
# Looking for the best number of clusters – I

- Some algorithms, such as K-means, require the number of clusters as a parameter
- Measures, such as SSE and Silhouette, are obviously influenced by the number of clusters
  - they can be used to optimize $K$
- Computation of Silhouette score is expensive
- SSE decreases monotonically for increasing $K$
  - is equal to TSS for $K = 1$
  - goes to zero when $K = N$

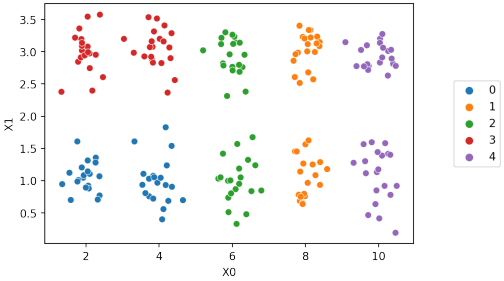
# Looking for the best number of clusters – II

- The *inertia* varying $K$ has frequently one or more points where the slope decreases: one of this points is frequently a plausible value for $K$
  - this is called elbow method
- The *silhouette* score varying $K$ has frequently a maximum, in this case it indicates the best value for $K$

# K–means results on the dataset of page 63



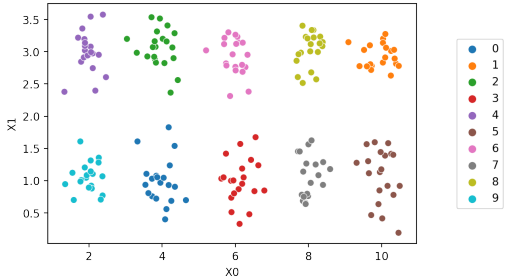

$K = 2$

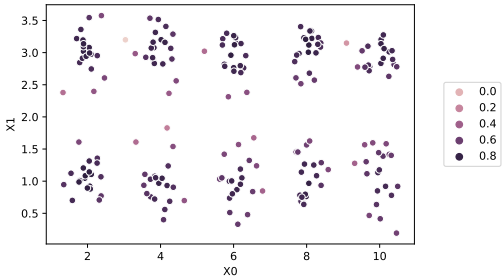



$K = 5$

# K–means results on the dataset of page 63

Best *silhouette for* $K = 10$



# Supervised measures: Gold Standard I

- Let be available a partition of a dataset similar to the data to be clustered, which we call gold standard, and defined by a labelling scheme $y_g(.)$
  - it is the same as the labels attached to supervised data for training a classifier
- Consider a clustering scheme $y_k(.)$
  - the cardinalities of the sets of distinct labels generated by the two schemes $\mathcal{V}_g$ and $\mathcal{V}_k$ can be different, and also in case of identity of the two grouping schemes, a permutation of labels could be necessary to make them equal

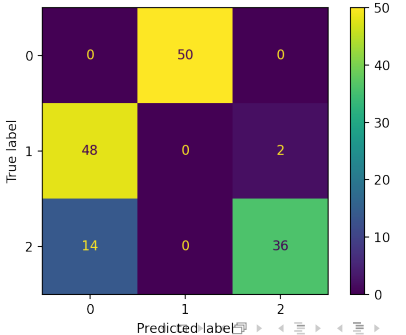
# Why should we compare it with the gold standard?

- validate a clustering technique which can be applied later to new, unlabelled data
- the purpose is quite similar to testing a classifier
- the difference is that in this case we are more interested in *grouping* new data than in *labelling them* following the Gold Standard scheme

# Classification–oriented measures

- Measure how the gold standard classes are distributed among the clusters
  - confusion matrix, precision, recall, f-measure
- On the right the confusion matrix for the Iris dataset
  - best match with permutation[1] of the predicted labels $0 \rightarrow 1, 1 \rightarrow 0, 2 \rightarrow 2$

```
X,y = load_iris(return_X_y=True)
estimator = KMeans(n_clusters=3
                   , random_state=363)
y_km = estimator.fit_predict(X)
disp = ConfusionMatrixDisplay(confusion_matrix(y,y_km))
disp.plot()
```



# Similarity oriented measures - I

- Analogous to compare binary data
- Consider a clustering scheme $y_k(.)$ and compare it with the Gold Standard $y_g(.)$
- Any pair of objects can be labelled as
  - *SGSK* if they belong to the same set in $y_g(.)$ and $y_k(.)$
  - *SGDK* if they belong to the same set in $y_g(.)$ and not in $y_k(.)$
  - *DGSK* if they belong to the same set in $y_k(.)$ but not in $y_g(.)$
  - *DGDK* if they belong to different sets both in $y_g(.)$ and $y_k(.)$

# Similarity oriented measures - II

Results given by
`pair_confusion_matrix(y_g,y_k)`

| | SK | DK |
|---|---|---|
| SG | 13512 | 1488 |
| DG | 1200 | 6150 |

---

Rand Score $\frac{SGSK+DGDK}{SGSK+DGDK+SGDK+DGSK} = 0.88$

Adjusted Rand Score Excludes the count of matches expected by chance[2] $= 0.73$

Jaccard Coefficient for label $c$ $\frac{SG_cSK_c}{SG_cSK_c+SG_cDK_c+DG_cSK_c} = (1, 0.75, 0.69)$

- it requires remapping of $y_g(.)$ to obtain the best match

2 See the Wikipedia page for a reference

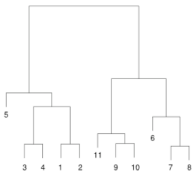
# Hierarchical clustering

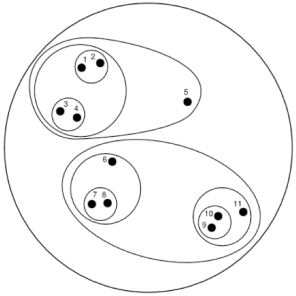
Generates a nested structure of clusters

- Agglomerative (bottom up)
  - as a starting state, each data point is a cluster
  - in each step the two less separated clusters are merged into one
  - a measure of separation between clusters is needed
- Divisive (top down)
  - as a starting state, the entire dataset is the only cluster
  - in each step, the cluster with the lowest cohesion is split
  - a measure of cluster cohesion and a split procedure are needed

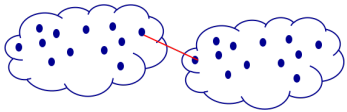
# Hierarchical clustering output

- Dendrogram (left)
- Nested cluster diagram (right)
- They represent the same structure
- The representation is the same for agglomerative and divisive
- The agglomerative methods are the most used



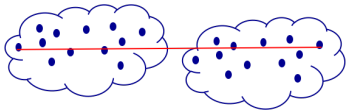



# Separation between clusters – Graph based



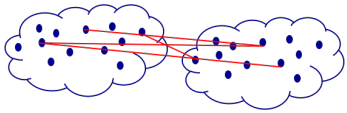
Single Link

$$\text{Sep}(k_i, k_j) = \min_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$



Complete Link

$$\text{Sep}(k_i, k_j) = \max_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$



Average Link

$$\text{Sep}(k_i, k_j) = \frac{1}{|k_i||k_j|} \sum_{x \in k_i, y \in k_j} \text{Dist}(x, y)$$

The distance between sets is based on the distances between objects belonging to the two sets, respectively

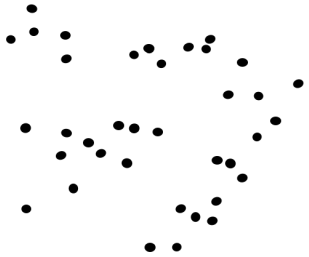
# Separation between clusters – Prototype based

Several alternatives, among them

- Distance between the centroids
- Ward's method
  - Given two sets with the respective SSE, the separation between the two is measured as the difference between the total SSE resulting in case of merge and the sum or the original SSE; smaller separation implies a lower increase in the SSE after merging

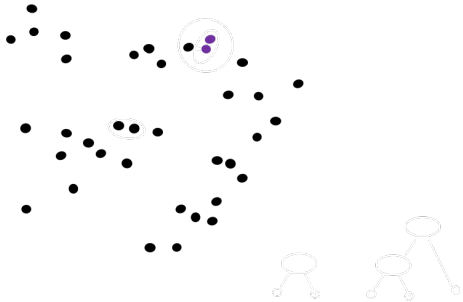
# Single linkage hierarchical clustering I

1. Initialization: every object is a cluster



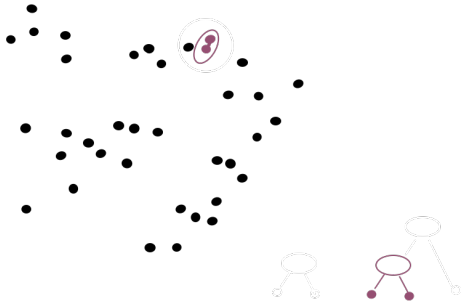
# Single linkage hierarchical clustering II

1. Initialization: every object is a cluster
2. Find the less separated pair



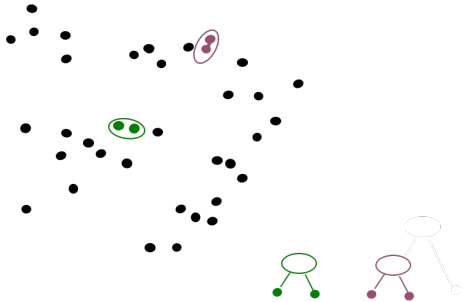
# Single linkage hierarchical clustering III

1. Initialization: every object is a cluster
2. Find the less separated pair
3. Merge in a single cluster



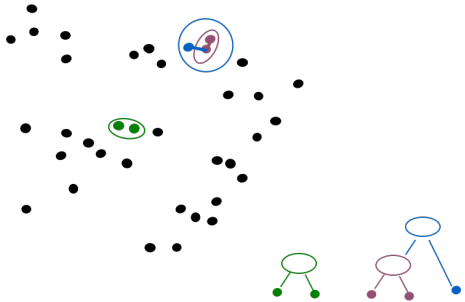
# Single linkage hierarchical clustering IV

1. Initialization: every object is a cluster
2. Find the less separated pair
3. Merge in a single cluster
4. Repeat



# Single linkage hierarchical clustering V

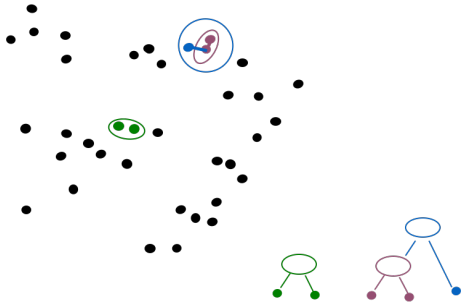
1. Initialization: every object is a cluster
2. Find the less separated pair
3. Merge in a single cluster
4. Repeat ...



# Single linkage hierarchical clustering VI

1. Initialization: every object is a cluster
2. Find the less separated pair
3. Merge in a single cluster
4. Repeat ...

The result is a dendrogram (taxonomy, object hierarchy)



# Single linkage algorithm

- Initialize the clusters, one for each objects
- Compute the distance matrix between the clusters, squared, symmetric, the size is the number of objects $N$, the main diagonal is null
- While the number of clusters is greater than 1
  - find the two clusters with lowest separation, say $k_r$ and $k_s$
  - merge them in a cluster
  - delete from the distance matrix the rows and columns $r$ and $s$ and insert one new row and column with the distances of the new cluster from the others

$$\text{Dist}(k_k, k_{(r+s)}) = \min(\text{Dist}(k_k, k_r), \text{Dist}(k_k, k_s)) \forall k \in [1, K]$$

# Time and space complexity

- Space and time: $\mathcal{O}(N^2)$ for the computation and the storage of the distance matrix
- Worst case $N-1$ iterations to reach the final single cluster
- For the i-th step of the main iteration:
  - search of the pair to merge $\mathcal{O}((N-i)^2)$
  - recomputation of the distance matrix $\mathcal{O}((N-i))$
- Time, in summary: $\mathcal{O}(N^3)$
  - can be reduced to $\mathcal{O}(N^2 \log(N))$ with indexing structures

# Italian cities example I

| | BA | FI | MI | NA | RM | TO |
|---|---|---|---|---|---|---|
| BA | 0 | 662 | 877 | 255 | 412 | 996 |
| FI | 662 | 0 | 295 | 468 | 268 | 400 |
| MI | 877 | 295 | 0 | 754 | 564 | 138 |
| NA | 255 | 468 | 754 | 0 | 219 | 869 |
| RM | 412 | 268 | 564 | 219 | 0 | 669 |
| TO | 996 | 400 | 138 | 869 | 669 | 0 |

# Italian cities example II

| | BA | FI | MI | NA | RM | TO |
|---|---|---|---|---|---|---|
| BA | 0 | 662 | 877 | 255 | 412 | 996 |
| FI | 662 | 0 | 295 | 468 | 268 | 400 |
| MI | 877 | 295 | 0 | 754 | 564 | 138 |
| NA | 255 | 468 | 754 | 0 | 219 | 869 |
| RM | 412 | 268 | 564 | 219 | 0 | 669 |
| TO | 996 | 400 | 138 | 869 | 669 | 0 |

# Italian cities example III

| | BA | FI | MI/TO | NA | RM |
|---|---|---|---|---|---|
| BA | 0 | 662 | 877 | 255 | 412 |
| FI | 662 | 0 | 295 | 468 | 268 |
| MI/TO | 877 | 295 | 0 | 754 | 564 |
| NA | 255 | 468 | 754 | 0 | 219 |
| RM | 412 | 268 | 564 | 219 | 0 |

# Italian cities example IV

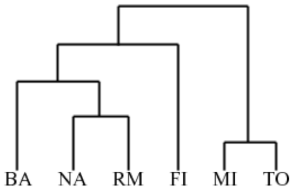
| | BA | FI | MI/TO | NA/RM |
|---|---|---|---|---|
| BA | 0 | 662 | 877 | 255 |
| FI | 662 | 0 | 295 | 268 |
| MI/TO | 877 | 295 | 0 | 564 |
| NA/RM | 255 | 268 | 564 | 0 |

# Italian cities example V

| | BA/NA/RM | FI | MI/TO |
|---|---|---|---|
| BA/NA/RM | 0 | 268 | 564 |
| FI | 268 | 0 | 295 |
| MI/TO | 564 | 295 | 0 |

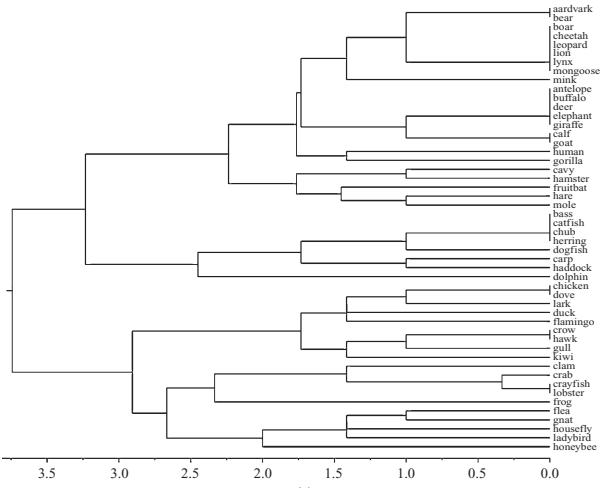
# Italian cities example VI

| | BA/FI/NA/RM | MI/TO |
|---|---|---|
| BA/FI/NA/RM | 0 | 295 |
| MI/TO | 295 | 0 |



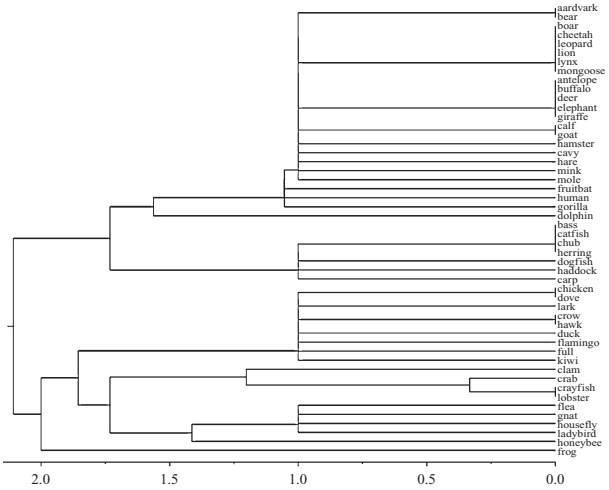

# The animals example – Complete Linkage

50 animals described with a numeric attribute (number of legs, scaled to [0,1]) and 15 boolean attributes, such as *has feathers, lays eggs, . . .*



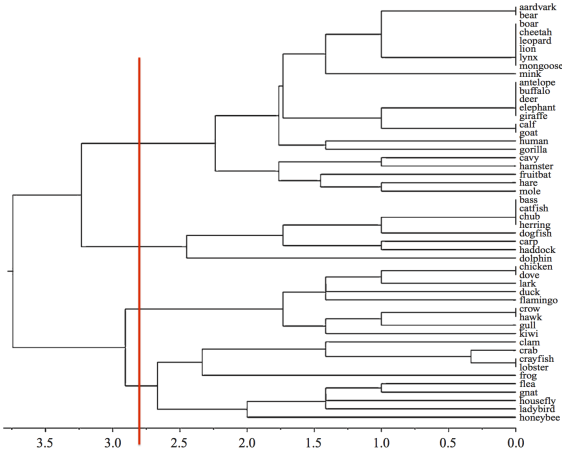
# The animals example – Single Linkage

50 animals described with a numeric attribute (number of legs, scaled to [0,1]) and 15 boolean attributes, such as *has feathers, lays eggs, . . .*



# Generating the clustering scheme

- The desired clustering scheme is obtained by cutting the dendrogram at some level
- The choice of the level is application dependent, and can also be guided by indexes, as in the case of K-means



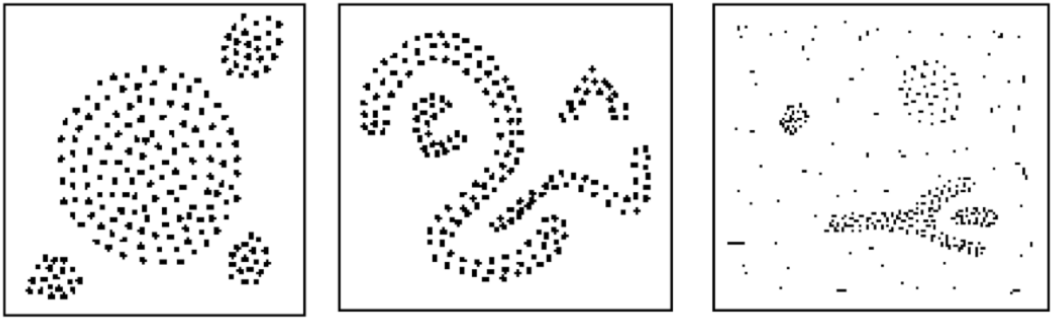
# Discussion I

- The horizontal axis in the dendrogram is the total dissimilarity inside the clusters, which obviously increases for decreasing number of clusters
- The diameter of a cluster is the distance among the most separated objects
  - Single linkage tends to generate clusters with larger diameters also at low levels
  - Complete linkage tends to generate more compact clusters

# Discussion II

- ☹ The scaling is poor, due to the high complexity
- ☹ There isn't a global objective function, the decision is always local and cannot be undone
- ☺ The dendrogram structure is of great help for the interpretation of the result
- ☺ Empirically, the result is frequently good

# Density based clustering



Clusters are high–density regions separated by low–density regions

# Computing density

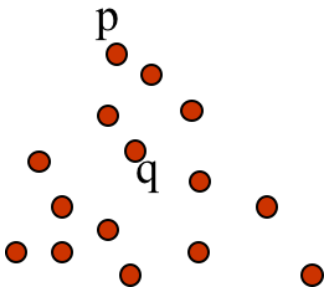
The two most obvious solutions

- Grid–based
  - split the (hyper)space into a regularly spaced grid
  - count the number of objects inside each grid element
- Object–centered
  - define the radius of a (hyper)sphere
  - attach to each object the number of objects which are inside that sphere

# DBSCAN – Density Based Spatial Clustering of Applications with Noise[3]

## Intuition

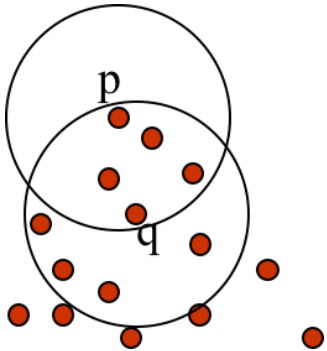
- *intuitively*, $p$ is a border point, while $q$ is a core point




3 [Ester et al.(1996)Ester, Kriegel, Sander, and Xu]

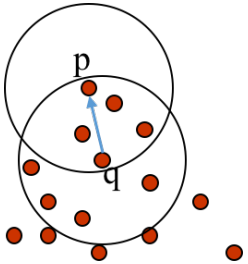
# Neighborhood

- define a radius $\epsilon$ and define as neighborhood of a point the $\epsilon$–hypersphere centered at that point
- points $p$ and $q$ are one in the neighborhood of the other
  - neighborhood is *symmetric*



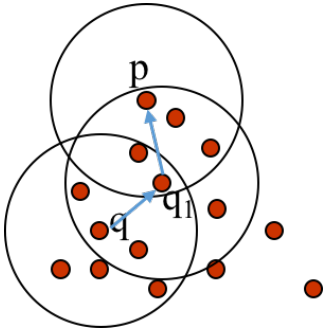
# Direct Density Reachability

- define a threshold $\mathrm{minPoints}$ and define as core a point with at least $\mathrm{minPoints}$ points in its neighborhood, as border otherwise
  - with $\mathrm{minPoints} = 5$, $q$ is core, $p$ is border
- define that a point $p$ is directly density reachable from point $q$ iff
  - $q$ is core
  - $q$ is in the neighborhood of $p$
- direct density reachability is not symmetric
  - in the example $q$ is not directly density reachable from $p$, since $p$ is border



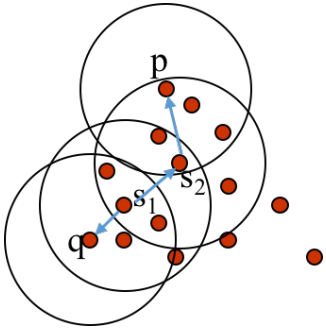
# Density Reachability

- a point $p$ is density reachable from point $q$ iff
  - $q$ is core
  - there is a sequence of points point $q_i$ such that $q_{i+1}$ is directly density reachable from $q_i, i \in [1, nq]$, $q_1$ is directly reachable from $q$ and $p$ is directly density reachable from $q_{nq}$
- reachability is not symmetric
  - in the example $q$ is not density reachable from $p$, since $p$ is border



# Density Connection

- a point $p$ is density connected to point $q$ iff there is a point $s$ such that $p$ and $q$ are density reachable from $s$
- density connection is symmetric



# Generation of clusters

- A *cluster* is a maximal set of points connected by *density*
- Border points which are not connected by density to any core point are labelled as noise

# Agorithm I

**Algorithm 1** DBSCAN

**Require:** SetOfPoints: UNCLASSIfIED points
**Require:** Eps, MinPts
  ClusterId ← nextId(NOISE);
  **for** i = 1 to SetOfPoints.size **do**
    Point ← SetOfPoints.get(i)
    **if** Point.ClId = UNCLASSIfIED **then**
      **if** ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) **then**
        ClusterId ← nextId(ClusterId)
**Ensure:** SetOfPoints

# Algorithm II

```
ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) : Boolean;
  seeds:=SetOfPoints.regionQuery(Point,Eps);
  If seeds.size<MinPts THEN // no core point
    SetOfPoint.changeClId(Point,NOISE);
    RETURN False;
  ELSE ...
```

# Algorithm III

```
    // all points in seeds are density-reachable from Point
     SetOfPoints.changeClIds(seeds,ClId);
     seeds.delete(Point);
     WHILE seeds <> Empty DO
       currentP := seeds.first();
       result := SetOfPoints.regionQuery(currentP,Eps);
       If result.size >= MinPts THEN
         For i FROM 1 TO result.size DO
           resultP := result.get(i);
           If resultP.ClId IN {UNCLASSIfIED, NOISE} THEN
             If resultP.ClId = UNCLASSIfIED THEN seeds.append(resultP);
             END If;
             SetOfPoints.changeClId(resultP,ClId);
           END If; // UNCLASSIfIED or NOISE
         END For;
       END If; // result.size >= MinPts
       seeds.delete(currentP);
     END WHILE; // seeds <> Empty
     RETURN True;
   END If
END; // ExpandCluster
```

# How to set $\epsilon$ and $\mathrm{minPoints}$?

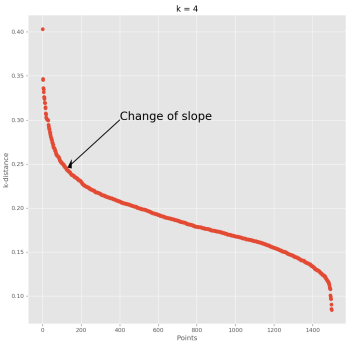
- As in many other machine learning algorithms, a *grid search* over several combination of hyperparameters can be useful
- As a *rule of thumb*, you can try $\mathrm{minPoints} = 2 * D$, the number of dimensions
- Noise suggest an increase in $\mathrm{minPoints}$
- A guess for $\epsilon$ requires more effort, considering the distance of the k–nearest neighbour, with $k = \mathrm{minPoints}$

# Good guess for $\epsilon$ I

- Consider the vector of the *k-distances*
  - choose $k$
  - for each point we compute the distance of its k–nearest neighbour and we sort the points for decreasing k–distance
- Choosing a given k–distance as $\epsilon$, it turns out that all the points with a k–distance bigger than $\epsilon$ will be considered as *border*
  - in the figure of next page they are the points to the left of the vertical of the chosen $\epsilon$

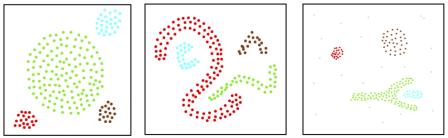
# Good guess for $\epsilon$ II

- Usually, datasets which exhibit some tendency to clustering exhibit also a *change of slope*
- The best $\epsilon$ can be found with a grid search in the *area of the change of slope*
  - the figure refers to a dataset with 1500 points and with $\mathrm{minPoints}$=4
  - this figure suggests a fine tuning of $\epsilon$ in the interval 0.2–0.3



# Comments

- ☺ Finds clusters of any shape
- ☺ Is robust w.r.t. noise
- ☹ Problems if clusters have widely varying densities
- Being based on distances between points, the complexity is $\mathcal{O}(N^2)$
  - reduced to $\mathcal{O}(N \log(N))$ if spatial indexes, such as R*, are available
- Very sensitive to the values of $\epsilon$ and $\mathrm{minPoints}$
- Decreasing $\epsilon$ and increasing $\mathrm{minPoints}$ reduces the cluster size and increases the number of noise points

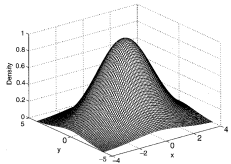


# Kernel Density Estimation [4]

- A technique developed in statistics and pattern mining
- Describe the distribution of the data by a function
- The overall density function is the sum of the influence functions (or kernel functions) associated with each point
- The kernel function
  - must be symmetric and monotonically decreasing
  - usually has a *parameter* to set the decreasing rate

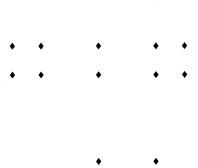
4 [Tan et al.(2006)Tan, Steinbach, and Kumar], Chapter 9

# KDE – Example

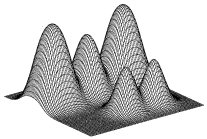



Gaussian kernel

Overall density – Grey scale plot



Sample set of 12 points



Overall density – Surface plot

# DENCLUE algorithm

1. Derive a density function for the space occupied by the data points
2. Identify the points that are local maxima
3. Associate each point with a density attractor by moving in the directions of maximum increase in density
4. Define clusters consisting of points associated with a particular density attractor
5. Discard clusters whose density attractor has a density less than a user–specified threshold $\xi$
6. Combine clusters that are connected by a path of points that all have a density of $\xi$ or higher

# DENCLUE comments

- ☺ It has a strong theoretical foundation on statistics
  - precise computation of density
  - DBSCAN is a special case of DENCLUE where the influence is a step function
- ☺ Good at dealing with noise and clusters of different shapes and sizes
- ☹ expensive computation $\mathcal{O}(N^2)$
  - can be optimized with approximated *grid based* computation
- ☹ Troubles with high dimensional data and clusters with different densities

# Model based (or statistic based) clustering[5]

- Estimate the parameters of a statistical model to maximize the ability of the model to explain the data
- The main technique is to use the mixture models
  - view the data as a set of observation from a mixture of different probability distributions
- Usually, the base model is a multivariate normal
  - well-known, easy to work with, good results
- The estimation is usually done using the maximum likelihood
  - given a set of data $\mathcal{X}$, the probability of the data, regarded as a function of the parameters, is called a likelihood function
- Attributes are assumed to be random independent variables

5 [Tan et al.(2006)Tan, Steinbach, and Kumar], Section 9.2.2

# Gaussian Mixture
# a.k.a. Expectation Maximization – EM

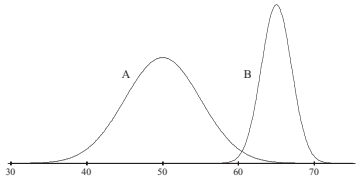
- If the data can be approximated by a single distribution, the derivation of the parameters is straightforward
- In the general case, with many mixed distributions, the EM algorithm is used

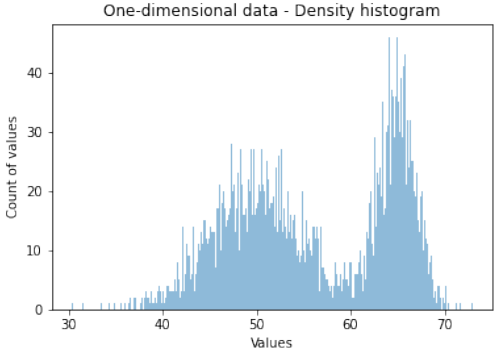
# EM algorithm

1. Select an initial set of model parameters
2. **repeat**
   - 2.1 **Expectation Step** – For each object, calculate the probability that each object belongs to each distribution
   - 2.2 **Maximization Step** – Given the probabilities from the expectation step, find the new estimates of the parameters that maximize the expected likelihood
3. **until** – the parameters do not change (or the change is below a specified threshold)

# One dimension mixture example

- Case with one dimension, two components
- Synthetic data randomly generated with two gaussians
  $\mu_A = 50, \sigma_A = 5, p_A = 0.6$
  $\mu_B = 65, \sigma_B = 2, p_B = 0.4$



# EM – one dimension, two clusters example I

- Need to estimate 5 parameters
  - mean and standard deviation for cluster A
  - mean and standard deviation for cluster B
  - sampling probability $p$ for cluster A

$$\mathbf{Pr}(A \mid x) = \frac{\mathbf{Pr}(x \mid A)\,\mathbf{Pr}(A)}{\mathbf{Pr}(x)} = \frac{f(x; \mu_A, \sigma_A)p_A}{\mathbf{Pr}(x)}$$

$$f(x; \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{\frac{(x-\mu)^2}{2\sigma^2}}$$

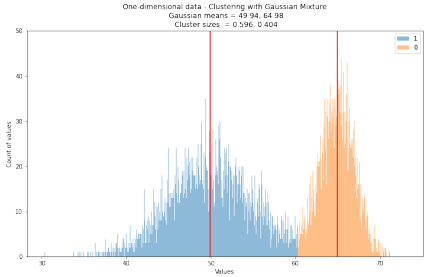
# EM – one dimension, two clusters example II

- Repeat until convergence
  - Expectation: Compute $p_A$ and $p_B$ using the current distribution parameters
    - Compute the numerators for $\mathbf{Pr}(A \mid x)$ and $\mathbf{Pr}(B \mid x)$ and normalize dividing by their sum
  - Maximization of the distribution likelihood given the data
    - Compute the new distributions parameters, weighting the probabilities according to the current distribution parameters
- After convergence label each object with $A$ or $B$ according to the maximum probability, given the last distribution parameters

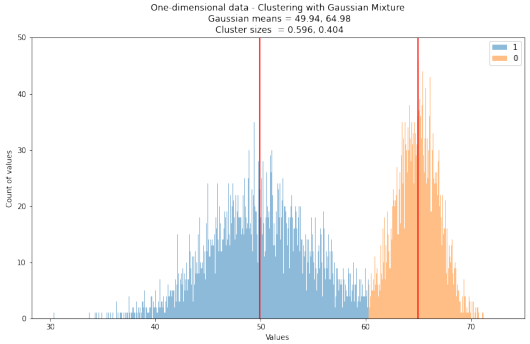
# One dimension example - Gaussian Mixture result

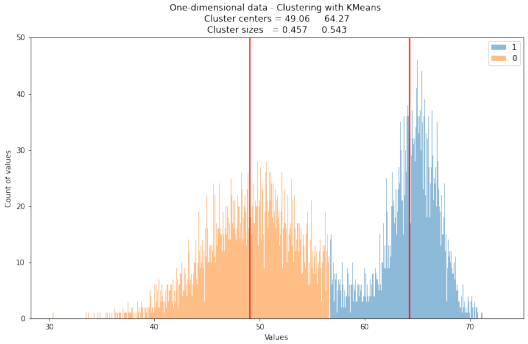
Estimated parameters of the distributions

| | weight | mean | deviation |
|---|---|---|---|
| 0 | 0.4037 | 64.978754 | 2.030968 |
| 1 | 0.5963 | 49.939990 | 4.999235 |



# Gaussian Mixture and KMeans – Comparison of results

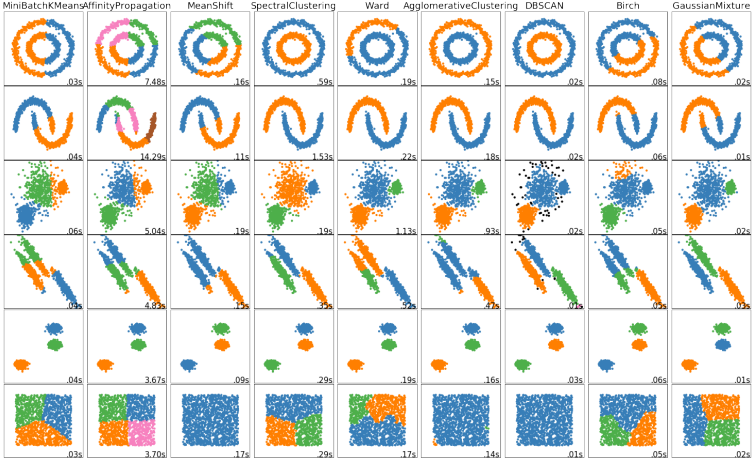



- These data have a *bimodal, gaussian–like* distribution
- The EM algorithm is founded on the hypothesis of modelling data with gaussians
- KMeans is *non–parametric*, and in this case the performance is *worse*

# Comparison of results for selected algorithms

(from Scikit-Learn documentation)



# A summary of selected clustering algorithms – I

| Method name | Parameters | Scalability | Usecase | Geometry (metric used) |
|---|---|---|---|---|
| K-Means | number of clusters | Very large n_samples, medium n_clusters with MiniBatch code | General-purpose, even cluster size, flat geometry, not too many clusters, inductive | Distances between points |
| Affinity propagation | damping, sample preference | Not scalable with n_samples | Many clusters, uneven cluster size, non-flat geometry, inductive | Graph distance (e.g. nearest-neighbor graph) |
| Mean-shift | bandwidth | Not scalable with n_samples | Many clusters, uneven cluster size, non-flat geometry, inductive | Distances between points |
| Spectral clustering | number of clusters | Medium n_samples, small n_clusters | Few clusters, even cluster size, non-flat geometry, transductive | Graph distance (e.g. nearest-neighbor graph) |
| Ward hierarchical clustering | number of clusters or distance threshold | Large n_samples and n_clusters | Many clusters, possibly connectivity constraints, transductive | Distances between points |

5 Find here a short discussion on inductive vs transductive learning

# A summary of selected clustering algorithms – II

| Method name | Parameters | Scalability | Usecase | Geometry (metric used) |
|---|---|---|---|---|
| Agglomerative clustering | number of clusters or distance threshold, linkage type, distance | Large n_samples and n_clusters | Many clusters, possibly connectivity constraints, non Euclidean distances, transductive | Any pairwise distance |
| DBSCAN | neighborhood size | Very large n_samples, medium n_clusters | Non-flat geometry, uneven cluster sizes, outlier removal, transductive | Distances between nearest points |
| OPTICS | minimum cluster membership | Very large n_samples, large n_clusters | Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive | Distances between points |
| Gaussian mixtures | many | Not scalable | Flat geometry, good for density estimation, inductive | Mahalanobis distances to centers |
| BIRCH | branching factor, threshold, optional global clusterer. | Large n_clusters and n_samples | Large dataset, outlier removal, data reduction, inductive | Euclidean distance between points |

# Clustering types

- Partitioning
  - iteratively find partitions in the dataset, optimizing some quality criterion
- Hierarchic
  - recursively compute a structured hierarchy of subsets
- Density based
  - compute densities and aggregates clusters in high density areas
- Model based
  - assume a model for the distribution of the data and find the model parameters which guarantee the best fitting to the data

# Clustering scalability

- Effectiveness decreases with
  - dimensionality $D$
  - noise level
- Computational cost increases with
  - dataset size $N$, at least linearly
  - dimensionality $D$

# Research perspective

- From the past
  - well-known problem in statistics
  - recent research
    - machine learning
    - databases
    - visualization
- For the future
  - effective and efficient algorithms for big data clustering, with a large number of dimensions, noise requested scalability w.r.t.:
    - size ($N$)
    - dimensionality ($D$)
    - noise level
    - rate of new data arrivals

# Uses of clustering – data comprehension

- Biology
  - Creation of taxonomies
  - Genetics
- Information Retrieval
  - Grouping documents
- Climatology
  - Repetition patterns
- Psychology and medicine
  - Identification of illness types in front of partial variation of evidence
- Business
  - Customer grouping

# Uses of clustering – utilities

- Summarization
  - Reasoning with groups representatives instead of with the entire population
- Data compression
  - Reduce the amount of data
  - Find cluster prototypes and substitute data with the indexes of the prototypes
    - Vector quantization
- Find the nearest neighbours
  - Each object refers to his prototypes
  - Near neighbours refer to the same prototype

# Bibliography I


- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu.
  A density-based algorithm for discovering clusters in large spatial databases with noise.
  pages 226–231. AAAI Press, 1996.
- A. K. Jain, M. N. Murty, and P. J. Flynn.
  Data clustering: A review.
  *ACM Comput. Surv.*, 31(3):264–323, September 1999.
  ISSN 0360-0300.
  doi: 10.1145/331499.331504.
  URL http://doi.acm.org/10.1145/331499.331504.

# Bibliography II


- Pang-Nin Tan, Michael Steinbach, and Vipin Kumar.
  *Data Mining*.
  Addison Wesley, 2006.
  ISBN 0-321-32136-7.